Aineiston esikäsittely klusterointia varten

Johannes Ukkonen
Johannes.Ukkonen@cs.helsinki.fi

Klusterointimenetelmät-seminaari
Helsingin yliopisto, tietojenkäsittelytieteen laitos
Raportti C–2002–54, s. 32–40, marraskuu 2002

Tiivistelmä

Aineiston esikäsittelyä tehdään, jotta klusterointialgoritmin tehokkuus ja lopputulos paranisi. Usein aineiston muokkaaminen on välttämätöntä, jotta lopputulos olisi likimainkaan oikein. Klusterointi voidaan usein ajatella kaksivaiheisena tehtävänä: klusterien muodostamisena ja niiden nimeämisenä. Klusterointiin parhaat esikäsittelymenetelmät pystyvät parantamaan kummankin tehtävän laatua.

Esikäsittely voi olla täysin klusterointialgoritmista erillinen tehtävä, joka tehdään kokonaan ennen varsinaista klusterointia, ja joka ei käytä mitään klusterointialgoritmin antamaa informaatiota hyväkseen. Joskus taas esikäsittelymenetelmä voi käyttää myös varsinaista klusterointialgoritmia tai sen osaa toistuvasti hyväkseen esikäsittelyn aikana.

Karkeasti määriteltynä klusteroinnin tarkoituksena on jakaa aineisto ohjaamattomasti klusteriryppäisiin. Määritettäessä klusteroinnin esikäsittelymenetelmiä luonteva tavoite on, että esikäsittelymenetelmätkin toimisivat itsenäisesti ja toisaalta mahdollisimman datariippumattomasti. Tämän kaltaisia esikäsittelijöitä on kuitenkin varsin vähän. Piirteiden valintaongelmaan, jota tässä raportissa käsitellään varsin yksityiskohtaisesti, voidaan itsenäinen esikäsittelijä toteuttaa.

1 Johdanto

Aineistoa, tai tässä raportissa synonyymina dataa esikäsitellään, jotta varsinaiset klusterointialgoritmit toimisivat luotettavammin ja tehokkaammin. Onnistuneen esikäsittelyn myötä voidaan toisaalta klusteroida yhä suurempia datajoukkoja ja toisaalta saada varsinaisen klusteroinnin tuloksena entistä oikeellisempia klusteriryppäitä. Toisinaan klusterointi voidaan nähdä kaksivaiheisena tehtävänä: yhtäältä klusterien muokkaamisena ja toisaalta niiden nimeämisenä. Näin ajatellen on erilaisilla esikäsittelymenetelmillä erilaisia vaikutuksia. Suurin osa esikäsittelymenetelmistä keskittyy klusterien muokkaamisen helpottamiseen, mutta toisaalta monet korostavat myös nimeämisongelman tärkeyttä [Tal99]. Yleiskäyttöisimmät esikäsittelytavat helpottavat kumpaakin tehtävää.

Datan esikäsittely voidaan nähdä täysin erillisenä tehtävänä, joka tapahtuu ennen kuin varsinainen klusterointi alkaa. Tämän johdosta esikäsittelyyn käytettävän menetelmän ei tarvitse olla missään tekemisissä klusteroinnin, ohjaamattoman oppimisen tai edes minkään koneoppimisparadigman kanssa. Ikävä puoli tällaisissa menettelyissä on usein se, että vaikka valittu klusterointialgoritmi toimisi varsin erilaisillakin syötteillä, ei koko ketjua mielivaltaisen kokoisesta ja muotoisesta alkuaineistosta klusteroinnin päättymiseen voida automatisoida. Toisaalta edes aktiivisesti varsinaista klusterointialgoritmia hyödyntävät esikäsittelymenetelmät eivät useinkaan ole kovin itsenäisiä (vaikkakin mahdollisesti sisältöriippumattomia) toimijoita, vaan vaativat käyttäjän syötteitä ajon aikana.

Hyvin usein datajoukko, jota on tarkoitus klusteroida, sisältää sekä turhia, että epäsymmetrisesti painotettuja piirteitä [MoS02]. Mikäli aineistoa ei mitenkään käsitellä ennen klusterointia, ei klusteroinnin lopputuloksen voi olettaa olevan ainakaan optimaalinen. Tarve esikäsittelyyn on siis ilmeinen. Mikäli datajoukko tunnetaan ihmisen tai koneen toimesta ja toisaalta tämän kaltainen toiminta-alueen asiantuntija on aina ennen klusterointia käytettävissä, on esikäsittely käytännössä triviaalia. Yksinkertaisesti karsitaan turhat piirteet ja painotetaan sopivasti loput asiantuntijan avulla. Klusteroinnin tavoitteena on kuitenkin ohjaamattomasti jakaa datajoukko ryppäisiin. Siksi myös esikäsittelyssä tulisi tavoitella yleistyskykyistä, ilman erityistä asiantuntijaa tehtävää toimintaa. Sekä piirteiden valinta, että datan skaalaus voidaan tehdä ohjaamattoman oppimisen menetelmin. Molempia käsitellään tarkemmin myöhemmin.

Tässä raportissa esitellään luvussa 2 yleisellä tasolla muutamia yleisiä menetelmiä datan esikäsittelemiseksi klusterointia varten. Luvussa 3 esitellään esimerkkinä piirteiden valinnalla saavutettavia etuja hierarkkisessa klusteroinnissa. Luvussa 4 tutkitaan tarkemmin piirteiden valintaa datan esikäsittelymenetelmänä ohjaamattomassa oppimisessa, johon klusterointi yhtenä edustajana kuuluu. Luvussa 5 on johtopäätöksiä ja koetuloksia piirteiden valintamenetelmistä. Luvussa 6 on raportin yhteenveto.

2 Erilaisia esikäsittelymenetelmiä

Tekoälytutkimuksen keskeisimpänä tavoitteena on pitkään pidetty täysin itsenäisesti toimivan älykkään agentin rakentamista [Lan97]. Koska klusterointi on ohjaamatonta oppimista, tulisi tavoitteeseen pääsemiseksi myös esikäsittelyn tapahtua ohjaamattomasti. Kuitenkin huomattava osa, itse asiassa lähes kaikki koneoppimistutkimuksissa kehitetyt datan esikäsittelymenetelmät on suunnattu ohjattuun oppimiseen. Ohjaamattomassa oppimisessa ei sellaisenaan voi näitä menetelmiä yleensä soveltaa. Pahin ongelma syntyy luokkien puuttumisesta, sillä ohjatun oppimisen esiprosessointimenetelmät käyttävät lähes poikkeuksetta jossain vaiheessa datapisteitten luokka-arvoja hyväkseen.

Tässä luvussa esitetään yleisellä tasolla kolme tapaa valmistella dataa klusterointia varten. Nämä ovat piirteiden valinta, datan skaalaus ja datan indeksointi.

2.1 Datan indeksointi

Joidenkin klusterointialgoritmien suorituskyky on kiinni siitä, kuinka nopeasti haluttu data saadaan käsittelyyn. Tämä ongelma tulee vastaan erityisesti tilanteissa, joissa data jota halutaan käsitellä, on (suuressa) spatiaalisessa tietokannassa.

Tietokantakyselyä voidaan nopeuttaa indeksoimalla dataa esimerkiksi useampiuloitteisella indeksoinnilla. Tällöin voidaan hyödyntää monia useampitasoiseen indeksointiin kehitettyjä tietorakenteita kuten KD-, R- tai R*-puita. Hakumenetelmänä käytetään tällöin yleensä lähimmän naapurin hakua (engl. *nearest neighbors search*) [Ber02].

Datan indeksointi kärsii ulottuvuuksien kasvusta. Jo yli 20-uloitteisessa avaruudessa indeksointi on lähes hyödyntöntä [Ber02]. Yleistyskykyiseksi klusteroinnin esiprosessointimenetelmäksi indeksointi ei siis käy.

2.2 Datan skaalaus

Datan skaalauksella tarkoitetaan kahta asiaa. Ensinnäkin datan sisältämille eri attribuuteille voidaan asettaa erilaisia painoja. Esimerkiksi klusteroitaessa kuvia, attribuutteja voisivat olla väri, teksti, muoto ja paikka. Nämä muodostavat neljä erillistä attribuuttiosajoukkoa, jossa esimerkiksi yksittäisten datapisteiden väri- ja paikka-arvot ovat keskenään hyvin erilaisia. Tutkimalla osajoukkojen välisiä suhteita voidaan klusteroinnin tehoa parantaa. Tarkempana esimerkkinä skaalauksesta voisi olla tilanne, jossa halutaan klusteroida dataa, joka sisältää joukon ihmisen pituuksia ja painoja. Kuvitellaan, että painot on ilmoitettu kilogrammoina ja pituudet metreinä. On ilmeistä, että näitä kahta attribuuttia tulee käsitellä erilaisilla asteikoilla. Yhden kokonaisluvun muutos painossa on tässä tapauksessa mitätön, mutta pituudessa todennäköisesti johtaisi asteikolta ulos. Tätä ongelmaa käsitellään usein ulottuvuksien vähentämisen (engl. *dimensionality reduction*) yhteydessä. Vaihtoehtona skaalaukselle esitetään usein myös klusterointia ryhmissä, jossa klusterointia tehdään attribuuttiosajoukoilla.

Modha ja Spangler [MoS02] ovat kehittäneet menetelmän, jolla attribuuttien skaalausta voidaan tehdä ohjaamattoman oppimisen keinoin k:n keskipisteen menetelmään. Attribuuttiosajoukkoja skaalataan yleistetyn Fisherin suhteen avulla, joka pyrkii samanaikaisesti minimoimaan klusterin sisäistä poikkeavuutta ja maksimoimaan eri klusterien välistä poikkeavuutta. Tällä tavalla voidaan löytää lähes optimaaliset piirrepainot sekä numeerisille, että kategorisille attribuuteille.

Toisaalta skaalauksella tarkoitetaan joskus tilannetta, jossa datapisteet eivät ole yhdenmukaisia ja ovat eri tavoin painotettuja. Esimerkkinä voitaisiin ajatella tilannetta, jossa halutaan käyttää klusterointia kauppaketjun kauppojen myymien tuotteiden valintaan. Tällöin voitaisiin luokitella kauppaketjun kauppoja niiden myymien tuotteiden prosentuaalisten määrien mukaan. Toisaalta myyntimäärältään suurien kauppojen voidaan olettaa ottavan huomattavan määrän tuotteita, jotka eivät välttämättä muodosta isoa osuutta kyseisen kaupan kokonaismyynnistä, mutta kuitenkin suuren osan "harvinaisemman"tuotteen kokonaiskysynnästä. Tällöin kaupan yleinen myyntimäärä antaa uuden painon suuremmille kaupoille. Eräillä skaalausmenetelmillä klusterien keskuspisteeksi muodostuu kunkin tuotteen joko myyntimäärältään suurin tai prosentuaalisesti tuotetta eniten myyvä kauppa [Ber02]. Tätä menetelmää kutsutaan joskus tapauskohtaiseksi skaalaukseksi (engl. *case scaling*).

Datan skaalaus voi siis olla yleistyskykyinen esiprosessointimenetelmä klusterointiin. Mikäli aineisto sisältää paljon tarpeettomia piirteitä, tulisi nämä kuitenkin ensin karsia.

2.3 Piirteiden valinta

Usein klusterointia varten kerätyssä aineistossa on klusterointialgoritmin kannalta turhia attribuuteja. Koska useimmat klusterointialgoritmit käyttävät kaikkia mahdollisia piirrevektoreita, on selvää, että attribuuttien määrä vaikuttaa algoritmin suoritusaikaan. Pahimmillaan myös lopputulos on vääristynyt. Jos klusteroinnin tavoitteena on sekä klusterien muodostaminen että niiden nimeäminen, on piiirteiden valinta tällöin erityisen tärkeää, sillä klusterit nimetään käytännössä kaikkien mahdollisten piirteitten mukaan. Turhat piirteet olisi siis tärkeää karsia pois ennen varsinaista klusterointia.

Ajatellaan taas tilannetta, jossa haluttaisiin klusteroida datajoukkoa, jossa on ihmisten pituuksia ja painoja. Lisäksi tässä tapauksessa aineistossa olisi kolmas attribuutti, kunkin ihmisen hemoglobiini. Jos klusterointi suoritetaan esimerkiksi pelkästään käyttämällä pituuksia ja painoja (tarpeen vaatiessa sopivasti painotettuna) tai pelkästään käyttämällä hemoglobiinia, saadaan lopputuloksena varmasti hyvin erilaisia klusteriryppäitä. Piirteiden valintaa suorittavan esiprosessoijan tulee päätellä, mikä tai mitkä attribuuteista ovat haitallisia klusteroinnin lopputulosta silmällä pitäen.

Edellinen on hyvin pelkistetty esimerkki. Tuntuu ilmeiseltä, että vastaavassa tilanteessa tiedettäisiin varmasti, millä perusteella klusterointi halutaan tehdä. Aina ei kuitenkaan ole näin: joko ei haluta käyttää tominta-alueen asiantuntijaa tai mahdollisesti asiantuntijaa ei ole. Piirteiden valinnalla pyritään jo ennen varsinaista klusterointia päättelemään mitkä datajoukon attribuuteista ovat tarpeellisia ja mitkä epäolennaisia. On klusterointialgoritmista kiinni valitsisiko esiprosessoija hemoglobiini- vai pituus- ja painopiirteet edellisestä esimerkistä. Tähän ongelmaan paneudutaan syvällisemmin seuraavissa luvuissa.

3 Piirteiden valinnalla saavutettavia etuja hierarkkisessa klusteroinnissa

Hierarkkisessa klusteroinnissa [Leh02] voidaan erottaa (ainakin) kolme etua, joita piirteiden valinnalla voidaan saavuttaa [Tal99]. Moniin muihinkin klusterointimenetelmiin pätevät nämä samat hyödyt. Ensinnäkin piirteiden valinnalla voidaan tunnistaa tarpeettomat piirteet. Jos tarpeettomia piirteitä on paljon, johtaa tämä yleensä oppijan suoritustehon heikkenemiseen. Oppijan suoritustehon heikkeneminen taas johtaa yleisesti huonompilaatuisiin klustereihin. Toisena piirteiden valinnan etuna voidaan pitää suorituskyvyn tehostumista. Koska erityisesti hierarkisessa klusteroinnissa konstruoidaan puu, joka pitää sisällään kaikki piirteet, on selvää, että piirteiden määrä vaikuttaa ennustustehtävän kompleksisuuteen. Valitsemalla sopivat piirteet voidaan kompleksisuutta pienentää, heikentämättä lopputulosta. Nämä kaksi valintamenetelmää siis tehostavat varsinaista klusterointitehtävää, mutta myös kuvaustehtävään voidaan piirteiden valinnalla saada parannusta. Kolmantena piirteiden valinnalla saavutettavana etuna voidaan ajatella tulosten ymmärrettävyyden parantamista. Koska klusterin nimeämiseen lähes kaikissa klusterointimenetelmissä käytetään kaikkia mahdollisia piirteitä, jotka vaikuttavat jollain tavalla kyseiseen klusteriin, johtaa piirteiden valinta lyhyempiin ja sitä myöten ymmärrettävämpiin klusterikuvauksiin. Viimeisen kohdan oletus lyhyemmän kuvauksen ymmärrettävyydestä ei toki päde aina, vaan joskus pidempi kuvaus on parempi ratkaisu. Toisaalta logiikkapohjaisissa klusterointimenetelmissä, joissa piirteitä valitaan sääntöjä poistamalla, ei piirteiden (esi-)karsinnalla ole kuvauksen ymmärrettävyyteen vaikutusta [Tal99].

4 Menetelmiä piirteiden valintaan ohjaamattomassa oppimisessa

Koneoppimisen saralla on piirteiden valintaan kehitetty monia menetelmiä vuosien saatossa. Valitettavasti tässä(kin) tapauksessa suurin osa tutkimuksesta on suuntautunut ohjattuun oppimiseen, eikä sellaisenaan sovellu klusterointimenetelmien esikäsittelyyn. Ohjattuun oppimiseen kehitetyistä piirteiden valintamenettelyistä voidaan erottaa kaksi tyyppiä: "suodatinmalli" (engl. *filter model*) ja "kääremalli" (engl. *wrapper model*). Vastaavasti ohjaamattomaan oppimiseen on edellisistä johdettu kaksi erilaista lähestymistapaa. Ohjaamattomassa oppimisessa piirteiden valintaan voidaan käyttää joko palaute- (engl. *feedback*) tai sokeaa menetelmää (engl. *blind method*) [Tal99]. Tähän on vielä syytä huomauttaa, että kuvatut tavat ovat runkoja, joilla pyritään erottamaan kaksi erilaista lähestymistapaa. Mikään ei estä yhdistämästä näitä kahta menetelmää tarvittavalla tavalla.

4.1 Palautemenetelmä piirteiden valintaan

Ohjaamattoman oppimisen piirteiden valintamenetelmät on johdettu vastaavista ohjatun oppimisen menetelmistä. Tämän johdosta esitetään ensin ohjatun oppimisen kääremalli, josta palautemenetelmä on suoraan johdettu.

4.1.1 Kääremalli piirteiden valintaan

Ohjatussa oppimisessa kenties yksinkertaisin menetelmä piirteiden valitsemiseksi on seuraava. Oletetaan, että käytössä on luokiteltu datajoukko ja jokin koneoppimismenetelmä [DeR97].

- Valitse pieni osa datasta esimerkkijoukoksi.
- Valitse attribuuttijoukko ja tutki kuinka koneoppimismenetelmä luokittelee esimerkkijoukon.
- Iteroi uusin attribuutein ja tarvittaessa uudella esimerkkijoukolla, kunnes luokittelu on (tarpeeksi) stabiili.

Näin menetelmä karsii pois turhat attribuutit ja tarvittaessa painottaa parhaita. Kääremallissa tutkitaan ainoastaan sitä, kuinka koneoppimismenetelmä luokittelee osajoukon, jolloin puhdas numeerinen vertailu eri attribuuttien hyvyyden välillä kertoo sen, mitkä niistä kannattaa valita. On kuitenkin ilmeistä, ettei se sovellu sellaisenaan klusterointiin, sillä menetelmä hyödyntää olennaisesti datapisteiden luokkia, joita klusteroitavassa datassa ei ole.

4.1.2 Algoritmirunko palautemenetelmään

Palautemenetelmä on varsin suoraviivainen toteuttaa. Ideana on se, että varsinaista klusterointia suorittavaa algoritmia ajetaan piirteiden osajoukoilla ja arvioidaan kunkin osajoukon kohdalla, kuinka hyvin klusteroinnissa on onnistuttu. Tässä menetelmässä ei tietenkään voida tukeutua suoraan luokittelijan toimintaan kuten ohjatussa oppimisessa, vaan joitain muita mittareita on kehitettävä. On varottava myös klusterointialgoritmin ylisovittamista, jota voi helposti tapahtua, mikäli koko harjoitusdatan joukkoa iteroidaan toistuvasti algoritmin syötteenä.

Ottamalla huomioon edelliset seikat, voidaan luoda seuraava algoritmirunko palautemenetelmälle:

- Jaa harjoitusdatan joukko kahtia, toinen varsinaista harjoitusta, toinen validointia varten.
- Suorita varsinaista klusterointialgoritmia erilaisilla piirteiden osajoukoilla.
- Vertaa algoritmin tulosta validointijoukkoon ja arvioi (piirre-)osajoukon laatua.
- Karsi pois piirteet, jotka huonontavat tai eivät paranna klusterointialgoritmin toimivuutta.

Joskus tässä menetelmässä voi olla hyvä käyttää myös ristiinvalidointia (engl. *cross-validation*), joka tarkoittaa sitä, että validointi- ja testijoukkoja vaihdellaan piirteiden valinnan aikana.

Palautemenetelmässä etsintään piirreavaruudessa voidaan hyvin käyttää ahnetta askeltavaa menetelmää, jolloin jokaisessa päätöspisteessä lisätään tai poistetaan yksi piirre. Alkutilanteessa piirrejoukko voi olla tyhjä, jolloin siihen lisätään piirre, joka eniten parantaa tarkkuutta tai alussa piirrejoukko voi olla kaikki piirteet sisällään pitävä, jolloin siitä poistetaan häiritsevin piirre joka askeleella. Yksinkertaisin tapa arvioida piirreosajoukon laatua on laskea väärin klusteroitujen alkioiden määrä kunkin piirrejoukon kohdalla. Paras piirreosajoukko tuottaa pienimmän virheen.

Menetelmän heikkoutena on paitsi ylisovittamisen vaara, myös validointijoukon määritys. Validointijoukko voisi olla asiantuntijan muodostama, mutta tällöin ei esiprosessoija toimisi itsenäisesti. Jos taas validointijoukko muodostetaan ennakkoarvauksella, ei se välttämättä ole luotettava.

4.2 Sokea menetelmä piirteiden valintaan

Koska myös sokea menetelmä on johdettu ohjatun oppimsen menetelmästä, tarkemmin suodatinmallista, käsitellään suodatinmalli ensin lyhyesti. Verrattuna ohjatun oppimisen kääremalliin, ei suodatinmallissa suoraan käytetä luokittelualgoritmia. Kääremallissa luokittelualgoritmin voidaan ajatella olevan "musta laatikko", jolloin esiprosessoijan ei tarvitse tuntea luokittelijan toimijaa [Tal00]. Suodatinmallissa tilanne on toinen. Suodatinmallissa on jokin mitta, joka riippuu datan sisäisestä (engl. *intrinsic*) ominaisuudesta. Tyypillisesti tämä mitta on jokaisen piirteen korrelaatio luokkaan nähden, esimerkiksi perustuen etäisyyteen tai riippuvuuteen. Datan sisäiseksi ominaisuudeksi valitaan se, minkä uskotaan olevan merkityksellinen lopullisessa suorituksessa, mutta joka ei suoraan vaikuta luokitteljan luokittelukykyyn. Myös tämä menetelmä on sellaisenaan soveltumaton ohjaamattomaan oppimiseen, sillä on mahdotonta laskea mitään mittaa, kun luokka-arvoja ei ole.

Samalla periaatteella määritetään sokea menetelmä. Aivan kuten suodatinmallissa, jossa ei käytetä suoraan luokittelijan antamaa informaatiota hyväksi, ei vastaavasti sokeassa menetelmässä käytetä klusterointialgoritmin tarjoamaa informaatiota piirteiden valintaan. Sen sijaan sokeassa menetelmässä nojataan johonkin aiempaan oletukseen tärkeimmistä piirteistä. Ja jotta oletus voitaisiin saada edes jossain määrin oikeaksi, tarvitaan jokin piirteen merkitysmitta. Piirteen merkitysmitan avulla on siis tarkoitus määrittää kuinka tärkeä yksittäinen piirre on klusteroinnin kannalta.

4.2.1 Piirteen merkitysmitan määrittäminen sokeassa menetelmässä

Ohjatussa oppimisessa suodatinmalleissa käytetään merkitysmittana usein piirteen ennustuskykyä yksittäisen datapisteen luokka-arvosta. Ohjaamattomassa oppimisessa on keksittävä muunlaisia mittareita. Yksi ratkaisu voisi olla se, että valitaan suoraan jokin jonkun klusterointialgoritmin käyttämä mitta, sillä tällaisen mitan tulisi suorittaa ennustavaa klusterointia. Koska klusterointialgoritmi suorittaa luokittelua nimenomaan jokaisen objektin piirteiden perusteella, on silloin luokittelijan mitan valinta luonnollinen pohja yksittäisen piirteen merkittävyyden arviointiin.

COBWEB on käsitteellinen klusterointimenetelmä [Fis87, Vil02]. Siinä klusterien valintaan käytetään seuraavaa, *category utility* - mittaa:

$$\frac{\sum_k P(C_k) \sum_i \sum_j [P(A_i = V_{ij}|C_k)^2 - P(A_i = V_{ij})^2]}{n}$$

Tämä mitta arvioi annetun klusterin C_k ennustuskykyä n klusterin jaossa, mille tahansa piirrearvolle V_{ij}, kun ominaisuuden todennäköisyys kaikille alkioille $P(A_i = V_{ij})$, joka saadaan tekemättä yhtään klusteria, on annettu. Ideana on maksimoida klusterin sisäistä samankaltaisuutta $P(A_i = V_{ij}|C_k)$ ja minimoida samanaikaisesti samankaltaisuutta toisten klusterien kanssa $P(C_k|A_i = V_{ij})$. Lyhyenä esimerkkinä [Tal00], jos klusteri C_k muodostetaan annetun arvon V_{11} ympärille, saa klusteri korkean $P(A_1 = V_{11}|C_k)$ todennäköisyyden, sillä suuri osa klusterin alkioista sisältää saman arvon. Kääntäen, koska suurin osa muista objekteista sisältää eri arvon, tulee myös tuloksesta $P(C_k|A_1 = V_{11})$ iso. Mikäli V_{11} korreloi toisen arvon V_{21} kanssa, sisältää C_k:n alkioista suuri osa myös arvon V_{21}.

Tästä havainnosta seuraa, että piirteet, jotka eivät juurikaan korreloi toisten piirteiden kanssa, ovat tarpeettomia [Tal00]. Toisin sanottuna, niistä piirteistä, jotka ovat tarpeellisia klusteroijalle, voidaan johtaa paremmin muita piirteitä. Category utility -mitan johtoa [Fis87] mukaillen voidaan johtaa yksittäisen piirteen merkitysmitta [Tal99, Tal00]. Seuraava kaava kertoo odotettavan määrän oikein ennustettavia piirrearvoja piirteellä A_M, kun tiedetään piirteen A_i arvo V_{ij}:

$$E(A_M, A_i) = \sum_{j_M} P(A_M = V_{j_M}|A_i = V_{ij})^2$$

Tässä määritelmässä oletetaan todennäköisyyksien samuus (engl. *probability matching*), eli että pirrearvo arvataan todennäköisyydellä $P(A_M = V_{j_M}|A_i = V_{ij})$, jolloin arvaus on oikein samalla todennäköisyydellä [Fis87]. Mikäli mitään piirrearvoja ei tunneta, voidaan määrittää vastaava "odotusmäärä" oikein ennustettavia piirrearvoja, jotka voidaan johtaa piirteestä A_M seuraavasti:

$$E(A_M) = \sum_{j_M} P(A_M = V_{j_M})^2$$

Näin voidaan tarkastella kuinka tieto muista piirteistä parantaa piirteen A_M ennustuskykyä seuraavalla kaavalla:

$$\frac{\sum_i \sum_j P(A_i = V_{ij})(E(A_M, A_i) - E(A_M))}{|\{i|A_i \neq A_M\}|} \tag{1}$$

Vasemannpuoleisin tekijä antaa korkeampia arvoja todennäköisemmille piirteille ja nimittäjä keskiarvoittaa tuloksen muitten piirteitten arvojen suhteen. Tuloksena saadaan yksittäisen piirteen paino. Kaavan 1 tarvitsemat todennäköisyydet saadaan laskemalla ne suoraan datasta. Johdettu kaava on lähes identtinen category utility -mitan kanssa, käytännössä vain klusteri-informaatio (edellä C_k:t) on hävitetty. Koska edellinen kaava on riippumaton luokkaarvoista ja varsinaisen klusterointialgorimin tuloksesta, soveltuu se esikäsittelyyn, piirteiden valinnan merkitysmitaksi.

Jos ajatellaan aikaisempaa esimerkkiä datajoukosta sisältäen ihmisten pituuden, painon ja hemoglobiinin, voidaan olettaa, että pituuksien ja painojen arvot korreloivat varsin voimakkaasti keskenään. (Painavat ihmiset ovat usein pidempiä.) Vastaavasti hemoglobiini ei korreloi. Kaavalla 1 siis sekä paino- että pituuspiirteillä tulos olisi hemoglobiinipiirrettä suurempi.

4.2.2 Algoritmirunko sokean menetelmän implementointiin

Koska edellisessä luvussa on saatu merkitysmitta, joka on riippumaton varsinaisesta klusterointialgoritmista, voidaan suoraviivaisesti johtaa algoritmirunko sokeaan menetelmään. Nimittäin jäljellä on enää se, millä perusteella yksittäinen piirre valitaan tai jätetään valitsematta. Tähän voidaan käyttää useampaa tapaa. Suoraviivaisinta olisi järjestää piirteet painojensa mukaan ja valita k painavinta. Toinen (parempi) tapa on määritellä piirteen valinnan kynnysarvo τ, missä τ on väliltä]0,1[[Tal99]. Voidaan johtaa seuraava algoritmirunko piirteiden valinnalle:

- Laske paino (kaava 1) jokaiselle piirteelle.
- Järjestä piirteet niiden painojen mukaan, painavin ensin.

- Poista loput piirteet, kyseinen piirre mukaan lukien, jolla paino $weight(A) < \tau \cdot weight(A_1)$.
- Jäljelle jäänyt joukko on tarpeellisten piirteiden joukko.

Näin on saatu johdettua yleistyskykyinen aineiston esikäsittelymenetelmä klusterointia varten. Tämä menetelmä on riippumaton aineiston sisällöstä.

5 Johtopäätöksiä

Edellä esiteltiin kaksi menetelmää piirteiden valintaan ohjaamattomassa oppimisessa: palaute- ja sokea menetelmä. Kummankin menetelmän avulla pystytään tunnistamaan epäolennaisia piirteitä datajoukosta. Siksi seuraava havainto on ehkä hiukan yllättävä: Kumpikaan menetelmä ei paranna klusteroinnin lopputulosta empiiristen havaintojen perusteella, vaan itse asiassa hiukan heikentää klusteroinnin lopputulosta [Tal99]. Klusterin nimeämisongelmaan piirteiden karsinnalla on positiivinen vaikutus, mutta tämä on "vaihdettava" heikentyneeseen luokittelukykyyn. Täytyy myös muistaa, että nimeämisongelmassa oletetaan suoraan, että lyhyemmät nimet ovat automaattisesti selkeämpiä. Koska kumpikin menetelmä aidosti karsii piirteitä, seuraa ongelman ratkaisu suoraan. Kahdesta menetelmästä sokealla on saatu parempia lopputuloksia kuin palautemenetelmällä. Jollain datajoukoilla sokea on ylittänyt sopivilla kynnysarvoilla jopa ilman karsintaa tehdyn klusteroinnin tuloksen.

Mistä edellinen havainto voi johtua? Tuntuisihan ilmeiseltä, että jos datassa on kokonaan epäolennaisia, suorastaan vääriä piirteitä, niiden karsinnalla olisi positiivinen vaikutus. Myös havainnot ohjatussa oppimisessa tukevat edellistä väitettä. Myöskin aiemmin johdetun sokean menetelmän piirteen merkitysmitan toimivuus tuntuu kiistämättömältä. Toisaalta palautemenetelmä toteutetaan suoraviivaisesti varsinaista klusterointialgoritmia hyödyntämällä. Tästä päätellen tullaan mielenkiintoiseen johtopäätökseen, että myöskin epäolennaiset piirteet jossain määrin tulevat olennaisiksi ohjaamattomassa oppimisessa [Tal99]. Lisätutkimus lienee tarpeen aidosti epäolennaisten piirteiden löytämiseksi.

Jotta raportista ei jäisi kuvaa, että piirteiden valinta esiprosessina klusterointiin on täysin väärä menetelmä, on syytä mainita positiivisista puolista. Nimittäin empiirisillä havainnoilla on jopa 60 prosenttia piirteistä pystytty karsimaan ilman, että klusteroinnin lopputulos on merkittävästi huonontunut [Tal00]. Näin merkittävä piirteiden määrän karsinta tehostaa kiistatta varsinaisten klusterointialgoritmien suorituksia. Myös klusterien nimeämisongelmaan saadaan varmasti näin hyötyä.

6 Yhteenveto

Mikäli esikäsittelyn tavoitteena on yksittäisen datajoukon muokkaaminen yksittäistä klusterointitehtävää varten, on esikäsittely (lähes) triviaalia. Usein tavoitteena on kuitenkin, että esikäsittelymenetelmä on paitsi automaattinen, myös yleistyskykyinen erilaisille datajoukoille ja eri klusterointialgoritmeille.

Tässä raportissa on kuvattu menetelmiä aineiston esiprosessointiin klusterointia, tai yleisemmin, ohjaamatonta oppimista varten. Erityisesti on keskitytty piirteiden valintaongelmaan, sillä se voidaan toteuttaa yleistyskykyisenä, ohjaamattoman oppimisen päättelytehtävänä. Piirteiden valinnan avulla pyritään datajoukosta erottamaan klusteroinnin kannalta turhat tai haitalliset piirteet ennen varsinaista klusterointia. Piirteiden valintaa on tutkittu paljon ohjatun oppimisen yhteydessä, mutta ohjaamattomassa oppimisessa tutkimustuloksia on vähän, kaikki aivan viime vuosilta. Toki esimerkiksi ulottuvuuksien vähentämistä on tutkittu klusteroinnissa varsin paljon, mutta siinä on usein hiukan erilaiset lähtökohdat kuin piirteiden valinnassa. Piirteiden valinnassa ei niinkään ole tavoitteena aineiston ulottuvuuden radikaali pienentäminen, vaan epäolennaisten ja haitallisten piirteiden tunnistaminen ja karsiminen.

Piirteiden valinta ohjaamattomassa oppimisessa on vaikea tehtävä, erityisesti kun kyseessä on datan esikäsittely. Vastaavia mittoja, kuten luvussa 4.2.1 tarvitaan. Muitakin lähestymistapoja, kuten dynaamista piirteiden valintaa, jossa valintaa suoritetaan klusteroinnin aikana, on esitetty [Tal99]. Yleisesti ottaen tutkimuskenttä on tässä kontekstissa vielä varsin avoin.

Viitteet

[Ber02] Pavel Berkhin. Survey of clustering data mining techniques. Research paper, Accrue Software, 2002. `http://www.accrue.com/products/rp_cluster_review.ps`, 11.10.2002.

[DeR97] Mark Devaney, Ashvin Ram. Efficent feature selection in conceptual clustering. In *Machine Learning: Proceedings of the 14th International Conference*, pages 92-97, Nashville, TN, USA, July 8-12 1997.

[Fis87] Douglas Fisher. Knowledge acquisition via incremental conceptual clustering. *Machine Learning*, 2:139-172, 1987.

[Lan97] Pat Langley. Machine learning for intelligent systems. In *Proceedings of the 14th International Conference on Artificial Intelligence*, pages 763-769, Providence, Rhode Island, USA, July 27-31, 1997.

[Leh02] Kari Lehmussaari. Hierarkkinen klusterointi. Teoksessa *Klusterointimenetelmät-seminaari*, Raportti C-2002-54, Helsingin yliopisto, tietojenkäsittelytieteen laitos, marraskuu 2002.

[MoS02] Dharmendra Modha, Scott Spangler. Feature weighting in k-Means clustering. *Machine Learning.* (To appear.) `http://www.almaden.ibm.com/cs/people/dmodha/ml02.ps`, 13.11.2002.

[Tal99] Luis Talavera. Feature selection as a preprocessing step for hierarchial clustering. In *Proceedings of the 16th International Conference on Machine Learning, ICML99*, pages 389-397, Bled, Slovenia, June 27-30 1999.

[Tal00] Luis Talavera. Dependency-based feature selection for clustering symbolic data. *Intelligent Data Analysis*, 4:19-28, 2000.

[Vil02] Kim Viljanen. Käsitteellinen klusterointi. Teoksessa *Klusterointimenetelmät-seminaari*, Raportti C-2002-54, Helsingin yliopisto, tietojenkäsittelytieteen laitos, marraskuu 2002.